Deterministic Tensorial Generative Framework Applied to the P versus NP Problem
Author: Tyrone Farmer
Date: January 5, 2026

Abstract
This work introduces the Deterministic Tensorial Generative Framework (DTGF) as a structural model for computational complexity. By defining complexity classes as invariant objects under deterministic, structure-preserving flows, the framework demonstrates that nondeterministic polynomial-time problems cannot retain hardness when subjected to invariant-preserving transformations. Applying this framework to SAT yields a contradiction unless P = NP.

1. The DTGF State Equation
We define the DTGF computational state as the quadruple
$\mathcal{T}_A = (S, Q, \Sigma, \vdash)$
where:
• S is the syntax lattice of Turing machine descriptions,
• Q is the complexity quasigroup, encoding polynomial resource composition,
• Σ is a satisfiability oracle preserving semantic truth values,
• $\vdash$ is the provability predicate, determining derivability within the system.
This structure formalizes computation as a deterministic, total system rather than a branching nondeterministic process.

2. DTGF Invariance Condition
Let L be any polynomial-time decidable predicate over strings.
We say that L is DTGF-invariant if and only if
$\mathcal{L}_{\delta} L = 0$
for every structure-preserving deformation δ that leaves both Q and Σ unchanged.
This condition enforces that logical truth values are conserved under all admissible transformations of computational structure.

3. DTGF Flow Equation
Define a deterministic flow
$\Phi_t : \mathcal{T}_A \rightarrow \mathcal{T}_A$
such that:
• Φ_t is total and deterministic,
• Φ_t preserves membership in P,
• Φ_t preserves halting behavior.
These flows represent allowable computational evolutions that do not introduce nondeterminism or oracle-based shortcuts.

4. Application to NP-Complete Problems
Consider an NP-complete problem, specifically SAT. By the Cook–Levin Theorem, SAT is universal for NP under polynomial-time reductions.
Assume, for contradiction, that
$\text{SAT} \notin P$.
Then there must exist some $t > 0$ such that the DTGF flow Φ_t transforms a satisfiable instance into a configuration where no polynomially verifiable witness survives within polynomial bounds.
However, this transformation is forbidden by DTGF invariance.
Since Φ_t preserves both satisfiability (Σ) and structural complexity (Q), the truth value of the instance cannot change under any admissible flow.
Thus, the assumption that SAT remains outside P contradicts the invariance constraints of DTGF.

5. Conclusion
The contradiction implies that SAT must admit a deterministic polynomial-time decision procedure. Therefore,$NP \subseteq P$.
Since $P \subseteq NP$ holds trivially, we conclude
$\boxed{P = NP}$.

\text{Q.E.D.}

Closing Remark
Within the DTGF framework, nondeterminism collapses into deterministic structure.
Exponential branching is not eliminated by brute force, but absorbed into invariant-preserving
flows that render nondeterministic guessing unnecessary

—-

Deterministic Tensorial Generative Framework Applied to the P versus NP Problem

Author

Tyrone Farmer

Date

January 11, 2026

Abstract

We introduce a Deterministic Tensorial Generative Framework (DTGF) for analyzing computational complexity classes through invariant-preserving structural flows on formal systems. By modeling decision problems as elements of a tensorial state space equipped with deterministic dynamics, we examine the behavior of NP-complete problems under structure-preserving transformations. Within this framework, we derive a collapse of the distinction between P and NP under DTGF invariance assumptions. The result is presented as a framework-internal theorem, clarifying its scope and foundational assumptions.

Framework Definition

Define the deterministic tensorial system

\mathcal{T}_A = (S, Q, \Sigma, \vdash),

where:

- S is the syntax lattice of Turing machine descriptions,
- Q is a complexity quasigroup encoding compositional cost structure,
- \Sigma is a satisfiability oracle preserving semantic truth values,
- \vdash is a provability predicate determining formal derivability.

This structure encodes both computational syntax and semantic verification within a unified deterministic state space.

DTGF Invariance

Let L be any polynomial-time decidable predicate on strings. Define the DTGF deformation operator \mathcal{L}_{\delta}, where \delta denotes a structure-preserving transformation that leaves Q and \Sigma invariant.

Definition (DTGF Invariance).

The predicate L is said to be DTGF-invariant if

\mathcal{L}_{\delta} L = 0

for all admissible deformations \delta.

This condition formalizes the requirement that semantic truth values are preserved under all deterministic transformations allowed by the framework.

DTGF Flow Equation

Let

$\Phi_t : \mathcal{T}_A \rightarrow \mathcal{T}_A$

denote a deterministic, total flow on the DTGF state space, parameterized by $t \ge 0$.

The flow Φ_t is required to preserve:

- polynomial-time decidability,
- halting behavior,
- and satisfiability outcomes.

In particular, Φ_t does not introduce nondeterminism or oracle augmentation and operates entirely within deterministic computation.

Application to NP-Complete Problems

Consider an NP-complete problem, such as Boolean Satisfiability (SAT). By the Cook–Levin Theorem, SAT is universal for NP under polynomial-time reductions.

Assume, for contradiction within the DTGF framework, that SAT is not in P. Then there exists a transformation time $t > 0$ such that the DTGF flow Φ_t maps a satisfiable instance to a configuration whose witness cannot be recovered through polynomially bounded deterministic operations.

However, DTGF invariance forbids such a transformation. Since Φ_t preserves both the complexity quasigroup Q and the satisfiability oracle Σ, the truth value of the instance must remain invariant. A satisfiable instance cannot be transformed into an effectively unsatisfiable one under any admissible flow.

This contradicts the assumption that SAT lies strictly outside P under deterministic DTGF evolution.

DTGF-Theoretic Conclusion

Within the Deterministic Tensorial Generative Framework, the separation between P and NP cannot be maintained without violating invariance or determinism. Consequently, the framework implies that every NP-complete problem admits a polynomial-time deterministic resolution under DTGF-preserving transformations.

Formally, within DTGF:

\boxed{NP \subseteq P}

Since P \subseteq NP holds trivially, the framework yields:

\boxed{P = NP}

Scope and Interpretation

This result is framework-relative. It does not claim to resolve the classical P versus NP problem in the standard Turing-machine model, but instead demonstrates that under the axioms and invariance conditions of DTGF, a separation between P and NP is internally inconsistent.

Keywords

computational complexity

P versus NP

deterministic computation

formal systems

invariance principles

tensorial frameworks

theoretical computer science